AN APP TO USE TFGM OPEN DATA

A PROJECT OVERVIEW AND PLAN
SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2018

By
Jung Huang
(10159031)

Supervisor: Konstantin Korovin
School of Computer Science

Contents

Word Count: 3338

List of Tables

List of Figures

Declaration

No portion of the work referred to in this project overview and plan has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Chapter 1

Introduction

Open data is non-private and non-confidential data that can be free to use and redistribute. The aim of open data is to make data more transparent, it will be easier for the public to make decisions and suggestions about knowledge or government policies based on detailed information.[1] There are many kinds of open data, one of the most important forms is open government data, which can be used to learn more about how government works or carry out research. For example, data.gov.uk [2] is a UK Government project launched in 2009, provides a variety of datasets and raw data for the public. Transport for Greater Manchester (TfGM) is one of the services in data.gov.uk, which provides an API (Open Data Service Version 2.0) about the transport network in Greater Manchester. The goal of TfGM is to encourage the development of applications that provide more information for travelers to make smarter choices.

The aim of this project is to build an app that provide the useful information for travelers by using TfGM open data. Since transport is one of the most important factors for a trip plan, in order to provide the more practical feature for travelers, this app integrates transport data with attraction data to generate the trip plan recommendation base on users preference. Whilst there are many websites and mobile apps about tourism, most of them only provide generic information but not a precise trip plan for the user; therefore, tourists have to spend time on deciding where to go and what to do. This app provides the suggestions for users who plan their trips or have trouble making decisions.

1.1 Aims and Objectives

The raw data provided by the APIs is very difficult for users to understand; therefore, the aim of this project is to create a mobile app that integrates the data from multiple APIs to generate a trip plan and present it in a way that travelers find acceptable.

1.1.1 Research Questions

- What is a good way to integrate TfGM open data with other open data, such as google map open data?
- What kinds of features are necessary for a trip plan?
- What is a good way to present auto-generated trip detail as human-written?

1.1.2 Learning Objectives

- Research and select several APIs that would provide available information for a trip plan.
- Compare and choose distinctive mobile app platform and study the app architecture with appropriate UX design.
- Research and set some constraints to make trip plan reasonable.
- Study how to interact data with the map and research the algorithm for route optimization.
- Study Natural Language Generation to understand the way to transfer semi-structured data into human-written style summary.

1.1.3 Deliverable Objectives

- Build an android mobile app to show a precise trip plan for travelers.
- Extract data from APIs and filter with constraints to list places, transport, time and other details.
- Use an algorithm to find the shortest path and show the route on map.
- Use NLG tool to transfer data into natural language.

- Generate test suite by different conditions to see the performance of the app.
- Analyze and evaluate the results by user experience and report recommendations for future work.

1.2 Project Overview and Plan Structure

Chapter 1: Introduction- gives a generic introduction to the project, including aims and objectives.
Chapter 2: Background- describes a brief literature review.
Chapter 3: Research methodology- explains how the project goals will be achieved.
Chapter 4: Ethics and professional considerations - demonstrates the ethics considerations in this project.
Chapter 5: Risk consideration - describes the potential risks and identifies the possible mitigations.
Chapter 6: Project evaluation- presents how the evaluation and testing might be done.
Chapter 7: Planning- illustrates a Gantt chart integrated with the text, technical milestones identified.

Chapter 2

Background

This chapter describes the relevant background knowledge and techniques in this project, including a brief introduction of data source, a comparison of different types of mobile apps, an introduction of NLG, and various algorithms of route optimization.

2.1 Open Data

This app adopts open data as the data source, there are two main sources come from TfGM API provided by data.gov.uk and google places API provided by Google API Platform.

2.1.1 TfGM API

TfGM API [3] provides transport information of Greater Manchester, e.g. schedules, routes, and locations, etc. This data is real-time (2000 calls/minute) and semi-structured data which is provided as an Open Data Protocol Representational State Transfer Application Programming Interface (OData RESTful API) [4] that uses Hypertext Transfer Protocol (HTTP) requests to interact.

2.1.2 Google Places API

Google Places API [5] provides places information in JSON or XML format by using HTTP requests. The requests include place search, details, query autocomplete, etc. Google Places API provides an example [6] about searching restaurant which contains the word cruise within a 1500m radius of a location through HTTP request and the response content as a list of JSON.

2.2 Types of Mobile Apps

There are three kinds of mobile apps: native, hybrid and web. The major difference between them is that native apps are compiled on a device, whereas hybrid and web apps are webpages; however, hybrid apps can be distributed in an app store. Table 2.1 shows the comparison of these three apps, and native app is developed in this project, which will explain in Research Methodology (see 3.1.2).

2.2.1 Native App

Native apps are developed for a particular platform and compiled directly on the device. Each platform uses default language, which is iOS on Objective C or Swift, Android on Java, or Windows Phone on Net, and cannot be used on different platforms. The main benefit is high performance and good user experience as each platform providing specific features. The disadvantage is that it can only be developed for one platform by specific language and APIs at a time. [7][8]

2.2.2 Web App

Web app which is a customized website looks and feels like a native app, runs on a browser that users can access from any device whenever there is an internet connection. Web apps are usually written in HTML5, JavaScript or CSS for client-side and possibly Java, PHP for server-side. The advantages include minimum of device memory is required, and it can be accessible from any device which has a browser. The drawback is that developers can only use the APIs provided by browser rather than native APIs or the platform. [7][8]

2.2.3 Hybrid App

Hybrid app is a web app disguised in a native wrapper so that it can access to the features of the mobile device, such as camera or GPS. Hybrid apps are compiled by a third-party tool, such as Apache Cordova, transforming the web app into a native app. The advantage is that it is fast and easy to develop because of single code base for all platforms. On the other hand, hybrid app has worse performance than native app and there may be some design issues for running on the different platforms. [7][8]

	Native app	Hybrid app	Web app
Development cost	high	moderate	low
Performance	fast	moderate	slow
Distribution	App store		web
Device features	Wide access	Some access	Few access
Cross platforms	no	yes	
Maintenance	hard	easy	
Recommended for	Apps that will be developed for single platforms	Applications that need to be distributed as multi-platform	Applications with limited funds, resources or terms
	Apps with wide requirements due to capabilities of hybrid or web	Those apps that will be developed for App Stores	Apps that do not require App Stores
	Anything that require highly optimization level for stable work		Developed with HTML, CSS, JavaScript etc.
	Apps that need best native UI or best graphic animation		

Table 2.1: comparison of native, hybrid, web app [9]

2.3 Natural Language Generation

Natural Language Generation (NLG) is a subtype of Artificial Intelligence (AI) that aims to generate natural language, such as summaries, reports, from a machine-readable format. The simplest form of NLG is filling in a blank by the data from other source. Another form is to use template to display the output, which dynamic data is generated by predefined rules. The more complex form is to produce an insightful narrative by considering the context, domain, etc. that users can comprehend. *(Perera R, Nand P, 2017)*[10]

2.3.1 NLG Tasks

The entire process of NLG is complex and involves a number of criteria to render an output that looks natural; therefore, it has to be split into several tasks. The typical stages of NLG *(Reiter and Dale, 1997, 2000)*[11] are:

1. Content determination: Deciding which information to mention in the text.
2. Text structuring: Determining the overall hierarchy of information in the text.
3. Aggregation: Organizing the sentences to enhance the naturalness.
4. Lexicalization: Finding the appropriate words for the narrative.
5. Referring expression generation: Selecting the words and phrases to identify objects and regions.
6. Realization: Creating the suitable text according to the rules of grammar, morphology, etc.

2.3.2 NLG Tool

Realization is the subtask of NLG, which aims to create the actual words and phrases for the well-formed sentences by several rules. There are a number of open-source surface realizers available, including Simplenlg, KPML, and OpenCCG. Table 2.2 shows a brief comparison of these tools, and Simplenlg is used in this project, which will explain in Research Methodology (see 3.1.3).

- Simplenlg *(A. Gatt and E. Reiter., 2009)*[12] : a simple Java API which is intended to function as a realization engine, focused on limited grammatical coverage compared to the other tools but easy to learn and use.

- KPML *(Bateman, 1997)*[13] : the oldest realizer written in ANSI Common Lisp with the graphical interface, which offers a mature platform for large-scale multilingual grammar development and generation.
- OpenCCG *(White, 2006)*[14] : an open-source realizer written in Java which provides many appropriate parsing and realization services according to Mark Steedmans Combinatory Categorial Grammar (CCG) formalism.

	Simplenlg	KPML	OpenCCG
Language	JAVA	ANSI Common Lisp	JAVA
Learning difficulty	easy	hard	
Advantage	Easy to learn and use	Mature platform with graphical interface	Provide various parsing and realization services
Disadvantage	Provide limited functionality	Developed by unfamiliar language	Less learning resources can be accessible
Focused on	Simple realization service	Large-scale multilingual development	Parsing and realization services

Table 2.2: comparison of NLG tools

2.3.3 NLG Evaluation

There are many ways for evaluating NLG systems, such as task-based, human rating, and metrics. *(E. Reiter., 2010)*[15] Table 2.3 shows a brief comparison of these evaluations, and human rating is used in this project, which will explain in Project Evaluation (see 6.2).

- Task-based (extrinsic) evaluation: is to try a system in the real-world and see if it has the desired effect. This evaluation is time and effort consuming, and usually needs many subjects because there are lots of statistical noise, such as ethical challenges, technical issues.
- Human ratings evaluation: is to ask the subjects to use the system, and then give some feedbacks about the system. For NLG system, users are typically asked to rate the usefulness, accuracy, readability, and comments of the generated texts.

- Metrics evaluation: is to compare the generated texts against a set of reference texts by specific metric, such as BLEU, METEOR, and ROUGE. This type of evaluation is broadly used in machine translation and document summarization.

	Task-based	Human ratings	Metrics
Time and effort consuming	long	moderate	short
Advantage	Reflect the real-world effect	Broader insights about a system can be obtained	It is the quickest and cheapest way
Disadvantage	Sometimes it is difficult to evaluate the system in real-world	It may not evaluate a specific hypothesis	It may not reflect real-world effect
Example system	STOP (which produced smoking-cessation letters)	Babytalk BT-Nurse system (which generates nursing shift handover reports)	Translation system (which translate text or speech from one language to another)
Recommended for	When the time and effort is allowed, and a large number of subjects can be found	When it is impossible to carry out the task-based evaluation	When it is impossible to carry out the task-based evaluation or human ratings evaluation

Table 2.3: comparison of NLG evaluations

2.4 Algorithms for Route Optimization

Route plan is one of the key parts in a trip plan, the related algorithms are discussed as follows, including ACO algorithm, Dijkstra algorithm, and A* algorithm. All of them can deal with the route optimization but in different ways. ACO algorithm is used in this project, which will explain in Research Methodology (see 3.1.4).

2.4.1 Ant Colony Optimization (ACO) Algorithm

ACO algorithm is a heuristic algorithm based on the behavior of ants seeking a path between the nest and the food. Ants will leave the pheromones on the road when they

are moving, and the other ants will follow the path which contain stronger pheromones. After a period of time, it can be seen that the majority of ants follow the same path which shows it is the best way to the food. The aim of ACO algorithm is to solve the travelling salesman problem, in which the goal is to find the shortest round-trip to link a series of places. *(M. Dorigo, 1996)*[16]

2.4.2 Dijkstra Algorithm

Dijkstra algorithm is a graph search algorithm to deal with the shortest path problem. It produces a shortest path tree with the lowest cost between a start vertex and every other vertex in the graph. In this algorithm, all the nodes would be gone through, if the new path is shorter than the current shortest path, that current path will be replaced by the new path. Dijkstra algorithm is widely used in the network routing protocols and to solve the graph-related problems. *(E. W. Dijkstra, 1959)*[17]

2.4.3 A* Algorithm

A* algorithm is a generalization of Dijkstra's algorithm, which traverses the graph and follows a path of the lowest known path. In any node, if the path has a higher cost than another one, it will choose the lower-cost path instead of higher-cost one. The process will stop since the goal is reached. A* algorithm is usually used in path finding and graph traversal. It can be also used to solve the problem of parsing using stochastic grammars in NLP. *(Peter E. Hart, Nils J. Nilsson, Bertram Raphael, 1968)*[18]

Chapter 3

Research Methodology

This section presents the methodology for achieving the goals of the project, including the related resources be used in the project, activity diagram, component diagram, and the possible extended work.

3.1 Resource, Tool and Algorithm

This app uses TfGM API and google places API as data resources, android studio as development environment, SimpleNLG as realization tool, and ACO algorithm for route optimization. This section explains the reason and the implementation of using these resources.

3.1.1 Multiple APIs

Data source is very important for a trip plan; therefore, this project uses TfGM API to obtain the real-time transport information to support the trip plan, including time schedule, route plan about multiple types of transport. Google places API is used to retrieve suitable attractions for the trip plan, which includes lots of information, such as opening hour, rating, pictures, etc.

3.1.2 Android Studio

There are many types of apps *(see 2.2)*, in order to provide high quality of UX and performance, this app is developed as an Android native app; therefore, choosing an appropriate development environment will help developers accelerate the building process. Android Studio [19] is Android's official integrated development environment

(IDE), provides many features for building apps on Android device. It offers a rich code editing, debugging, testing, and support instant run, sample apps, layout editing, etc.

3.1.3 SimpleNLG

Transfer trip plan data into a paragraph is one of the important steps for this project. Compare to the other NLG tools *(see 2.3.2)*, SimpleNLG is a simple JAVA API which is easy to learn and use. It can generate a proper sentence by giving some words, like subjects, verbs, objects. Besides, both of SimpleNLG and Android app are based on JAVA, it would be more convenient to incorporate the library into the app.

3.1.4 Ant Colony Optimization Algorithm

A smart route plan can not only enhance the quality of the trip plan but also save time on transport; therefore, a suitable algorithm is needed for the path optimization. Compare the multiple algorithms *(see 2.4)*, considering the most of the trip plans are round-trip; therefore, this project will use the ACO algorithm to calculate the best route between the places and then choosing the appropriate transport types to implement it.

3.2 Design Diagrams

The aim of this project is to build a trip plan app. The following content presents the structure and the process design of the app.

3.2.1 Activity Diagram

Figure 3.1 shows the plan of activity design.

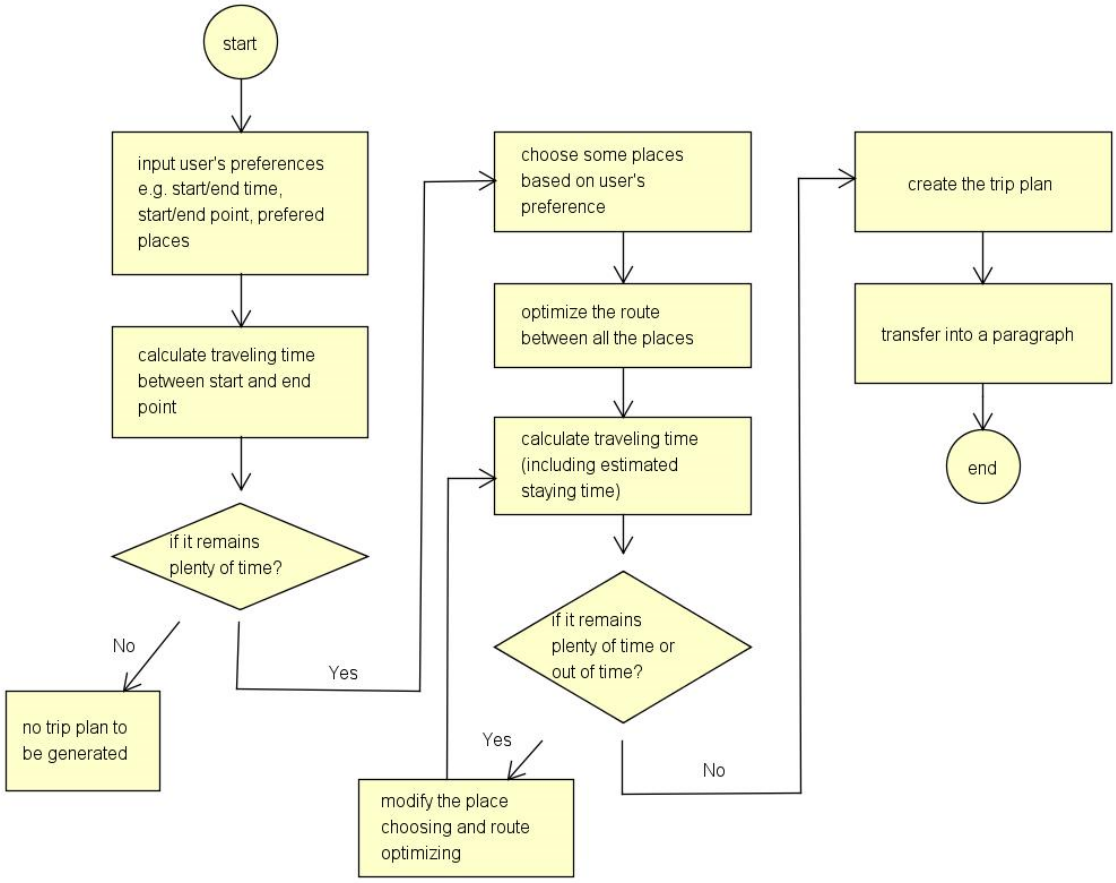

Figure 3.1: activity diagram

3.2.2 Component Diagram

Figure 3.2 shows the plan of component design.

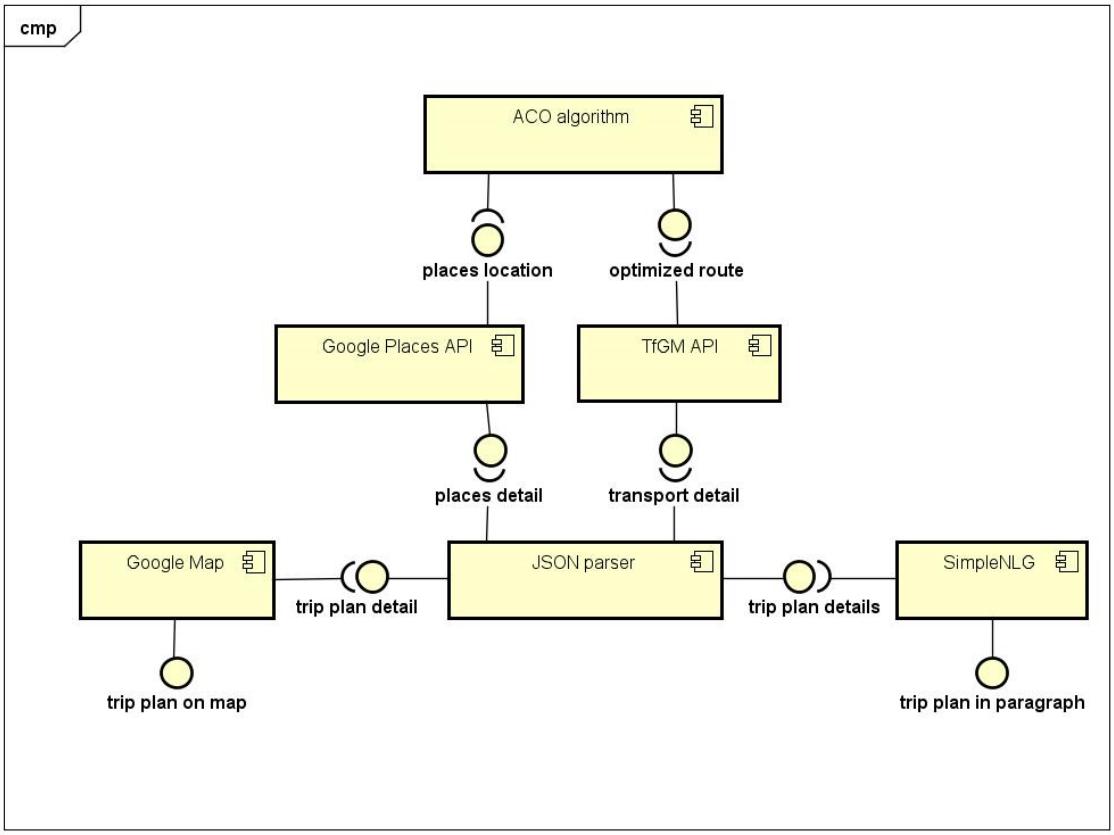

Figure 3.2: component diagram

3.3 Extended Work

This app would be done by randomly choosing several places base on the user preference; however, there are much more considerations can be involved. If time is allowed, this app can be enhanced quality by avoiding popular time of the place, selecting the places based on rating, letting user modify the staying time and places. Moreover, the trip plan can be varied by changing the description; for example, paraphrasing ”try the Italian gourmet in B” to ”satisfy your appetite in B”, the aim is to make it more natural (like human write).

Chapter 4

Ethics and Professional Considerations

There are two ethics and professional issues should be considered in this project. Firstly, abiding the rules when accessing the API; secondly, protecting the personal information when asking users to do the survey.

4.1 Accessing the API

It is vital to understand and abide by the APIs Rules when using their services. API providers would define several terms and conditions in the document, such as the licence of TfGM API follows the Terms of the Open Government Licence version 3.0 [20] (OGL v3.0) and the terms of google API is defined in Google APIs Terms of Service [21]. Overall, there are two main points. Firstly, the developers who would like to use the API service should provide the accurate personal contact information and accept the terms before accessing the API; secondly, acknowledge the source of the information in the application by including an attribution statement, such as ”Contains Transport for Greater Manchester data” for TfGM.

4.2 Doing Survey

This project is evaluated by human rating that users are asked to fill in a questionnaire. It is important to provide respondents with the purpose of survey, the possible of risks, a privacy statement, and an introduction about what kinds of questions will be asked

so that they can decide whether to participate or not; all the information should be voluntarily provided and the results should be faithfully presented. Besides, participants have authority to know who access the data and how it will be conducted. Moreover, any confidentiality or anonymity must be assured. [22]

Chapter 5

Risk Consideration

It is important to consider the potential risks before building the application, the following content describes the possible risks and mitigations in the different aspects: time estimation for planning the trip schedule, poor performance for developing the app, and losing data for accessing the API.

5.1 The risk of Time Estimation

The time spending on each place depends on individuals; therefore, it is difficult to estimate the staying time for the trip plan. Although Google Map App provides estimated visiting time, this information cannot be retrieved from the google places API. The possible solution is that make the trip plan more flexible. Users can generate a new trip plan if they finish the current trip early; however, it is impossible to modify the plan as it is out of time.

5.2 The risk of Poor Performance

There are some potential risks which make system work slowly; for example, the poor connection of internet would affect the speed of retrieving data from APIs; the query result may be too large to load; the process of realization may be complicated. In order to avoid these risks, the size of query result should be limited and use the appropriate algorithm to deal with route optimization and realization. Moreover, present the alert message when the internet connection is poor or any error occurs.

5.3 The risk of Losing API Access

API providers have authority to modify or discontinue their services, which may affect developers a lot. Consuming data from API is crucial to a trip plan app; therefore, risk manage about accessing the API should be considered. The possible way to mitigate the risk is that preparing a list of alternative APIs that provide the similar functionality as currently consuming one; for example, Google map could be replaced by OpenStreetMap. Besides, review the terms of service of the API frequently in case there is any change arises.

Chapter 6

Project Evaluation

This section describes the three testing stages and the test-driven development strategy as testing method, and the human rating evaluation as evaluation method.

6.1 Testing Method

The test for the mobile app includes three main categories: small, medium, and large. These categories are typically split into: 70 percent small, 20 percent medium, and 10 percent large. Small tests are the unit tests that aim to test each major component, medium tests are the integration tests to run the several components at the same time, and large tests are the UI tests that make sure all the user tasks work as expected. [23]

This project will adopt test-driven development (TDD) as testing strategy. The process of TDD is an iterative cycle, starts with a failed test because the feature is not implemented yet, the aim of it is to make sure the implementation fits the requirement correctly. [24] Besides, the full workflow of testing contains the three categories mentioned before; small tests start in the beginning of the development, medium tests start at the middle of stage, and large tests start at later period. The whole testing process continues until every use case is satisfied.

6.2 Evaluation Method

There are many ways to evaluate the NLG system *(see 2.3.3)*, the human rating evaluation is the best one for this app because the quality of this app is based on the users feeling. The target customer of the app is the travelers in Greater Manchester or someone who wants to travel there. They will be asked to fill in a questionnaire to rate the

usefulness, accuracy, readability, and comments of the app. The questions are asked to evaluate the generated summary, like ”Can you understand the narrative of the trip plan?”, ”Do you think the trip plan help you make decision?”.

In addition to the NLG part of the app, the other parts of the app can also be evaluated by human rating. Some questions are asked to understand the user satisfaction, like ”Will you follow the trip plan generated by this app?”, ”Do you think the trip plan route is reasonable?”. Moreover, some questions would be asked in order to improve the app, such as ”What functionality do you suggest containing in this app?”, ”Do you think which part of the app should be improved?”.

Chapter 7

Planning

Figure 7.1 illustrate a Gantt chart for the project plan, including the implement details, the deadlines, and the several technical milestones.

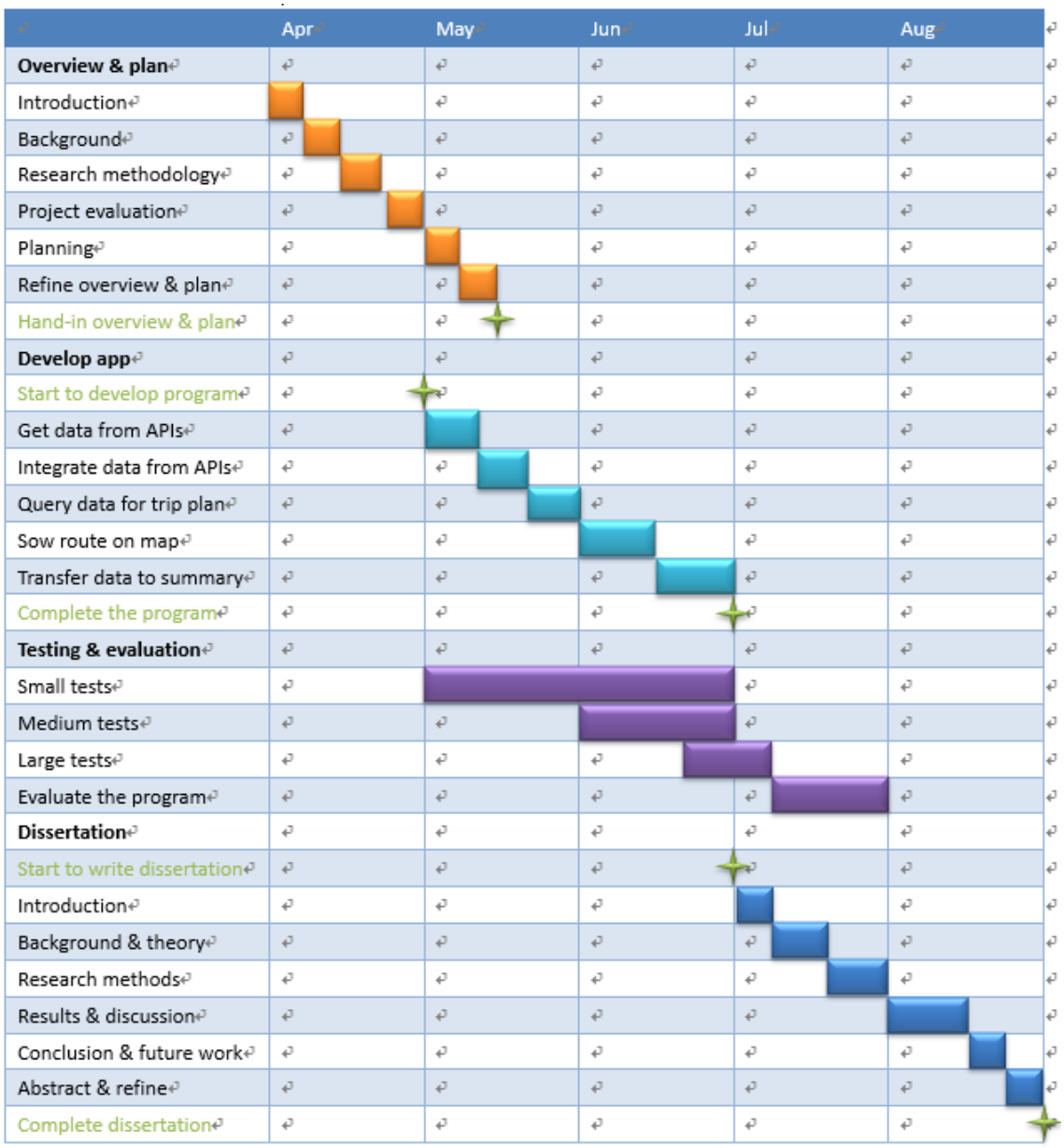

Figure 7.1: Gantt chart for the project plan

Bibliography

[1] M. Janssen, Y. Charalabidis, and A. Zuiderwijk, "Benefits, adoption barriers and myths of open data and open government," *Information Systems Management*, vol. 29, no. 4, pp. 258–268, 2012.

[2] data.gov.uk, "https://data.gov.uk/."

[3] T. for Greater Manchester, "https://developer.tfgm.com/."

[4] O. R. API, "http://www.odata.org/."

[5] G. P. API, "https://developers.google.com/places/web-service/intro."

[6] G. P. A. example, "https://developers.google.com/places/web-service/search."

[7] X. Zhiming, "Mobile app development," *Information Technology*, 2016.

[8] N. Serrano, J. Hernantes, and G. Gallardo, "Mobile web apps," *IEEE Software*, vol. 30, pp. 22–27, Sept 2013.

[9] ThinkMobiles, "https://thinkmobiles.com/blog/popular-types-of-apps/."

[10] N. P. Perera R, "Recent advances in natural language generation: A survey and classification of the empirical literature," *Computing and Informatics*, vol. 36, pp. 1–32, 2017.

[11] E. Dale, Robert; Reiter, "Building natural language generation systems," *Cambridge, U.K.: Cambridge University Press*, 2000.

[12] A. Gatt and E. Reiter, "Simplenlg: A realisation engine for practical applications," in *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, (Stroudsburg, PA, USA), pp. 90–93, Association for Computational Linguistics, 2009.

[13] J. A. Bateman, "Enabling technology for multilingual natural language generation: The kpml development environment," *Nat. Lang. Eng.*, vol. 3, pp. 15–55, Mar. 1997.

[14] M. White, "Ccg chart realization from disjunctive inputs," in *Proceedings of the Fourth International Natural Language Generation Conference*, INLG '06, (Stroudsburg, PA, USA), pp. 12–19, Association for Computational Linguistics, 2006.

[15] R. Ehud, *Natural Language Generation*, ch. 20, pp. 574–598. Wiley-Blackwell, 2010.

[16] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, pp. 29–41, Feb 1996.

[17] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, pp. 269–271, Dec. 1959.

[18] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, July 1968.

[19] A. Studio, "https://developer.android.com/studio/."

[20] O. G. Licence, "http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/."

[21] G. A. T. of Service, "https://developers.google.com/terms/?hl=zh-tw."

[22] J. S. House *The Public Opinion Quarterly*, vol. 56, no. 1, pp. 139–141, 1992.

[23] G. Developers, "https://developer.android.com/training/testing/fundamentals."

[24] D. Fucci, H. Erdogmus, B. Turhan, M. Oivo, and N. Juristo, "A dissection of the test-driven development process: Does it really matter to test-first or to test-last?," *IEEE Transactions on Software Engineering*, vol. 43, pp. 597–614, July 2017.